Dynamic Software License Configuration

Inventors

Yuri Leontiev, Michele Marik, Joseph Andrew Smith

Background

Field of Invention

[0001] The present invention relates generally to dynamically configuring user software licenses.

Background of Invention

[0002] Software piracy is very prevalent, and causes a significant loss of potential revenue for software vendors. For this reason, many software vendors require that a user activate a software license before operating a software program. Typically, the software vendor will provide a unique identifier with each copy of the software. In order to run the software, the user must provide that identifier to the vendor (either via the Internet, by telephone, or some other way). The vendor associates the identifier with the specific user, and stores a record of that association, typically in a database or the like. The vendor then activates the user's copy of the software, either automatically over the Internet, or by providing the user with an activation code to enter manually. Typically, activation involves updating a license file on the user's computer, to indicate that the software is activated. Whenever the software program runs, it will check the license file, and will only run if the software has been activated.

[0003] This system fails to prevent a user from making unauthorized copies of the software program, once the software program has been activated. Because the activation is performed once, and a record of the activation is kept on the user's computer in the form of the updated (activated) license file, the user can install the software program on a second user's computer, by simply copying the complete contents of the folders that hold the installed software program. Because this will copy the activated license file as well as the software program, the second user will now be able to run the software program. Such unauthorized copying is known is "pass along piracy."

[0004] One method used to prevent pass along piracy is to include a hardware profile of the user's computer in the license file. During the activation process, the software program can scan the hardware of the computer of the authorized user. Then, a profile of the hardware configuration can be stored in the activated license file. When the software program is subsequently run, it can not only check to see if an activated license file is present, but can also check to see if the hardware of the computer it is running on corresponds to that of the authorized user. The software program does this by scanning the hardware of the computer it is running on, and comparing the result to the hardware profile of the authorized user in the license file. Only if the hardware is the same or within an acceptable margin of difference will the software program run. That way, if the user installs an unauthorized copy of the software program on a second user's computer, when the second user tries to run the software program, the software program will determine that it is installed on an unauthorized computer and not run.

[0005] One problem with this methodology is that it does not allow an authorized user to upgrade his hardware. For example, if an authorized user changes

hardware components of his computer (e.g. changes the monitor, adds a cable modem, adds memory), the current hardware will no longer match the stored hardware profile and the software program will not run. Additionally, the same problem will occur if the authorized user purchases a new computer, and wishes to transfer the software program from the older computer to the new one.

[0006] Some software vendors allow a user to call on the telephone and inform the vendor of such changes. If the vendor is convinced that the user really has modified his hardware configuration or purchased a new computer, the vendor can activate the software on the new or modified computer. However, the vendor has no way of knowing whether the user has really made a legitimate hardware modification, or whether the user is attempting to make an unauthorized copy of the software. Suppose that the user has actually engaged in pass along piracy, by installing the software on a second user's computer. If the user convinces the vendor that the new installation is legitimate, the vendor will unknowingly activate the software on the second user's computer. Because the software has already been activated on the first user's computer, nothing stops the first user from continuing to run the software on his computer, while the second user runs the pirated software on a second computer. On the other hand, if the user has legitimately upgraded his computer but cannot convince the vendor, the user will not be able to run the software product at all.

[0007] Thus, existing software licensing activation methods either do not adequately prevent pass along piracy, or can prevent an authorized user from running the software product on his own computer, after making legitimate hardware modifications. It would be desirable for a software vendor to be able to dynamically extend or revoke

user software licenses, for example in order to transfer a software license from a first to a second computer, to extend a license to a legitimately upgraded computer, or to revoke a software licenses on a first computer after a transfer, or where piracy is detected.

[0008] Furthermore, existing license activation methodology does not allow vendors to make other dynamic changes to a user software license, such as upgrading or otherwise modifying the license. It would be desirable for software vendors to be able to make these types of dynamic modifications to user software licenses as well. Consequently, what is needed are methods, systems and computer readable media for dynamically configuring user software licenses.

Summary of Invention

[0009] In some embodiments, a user runs a client software program on a computer. As the user runs the software program, the user attempts to access various features of the software program. Requests to access features of the software program are transmitted to a server. Such requests include identification information and license verification information concerning the user.

[0010] The server maintains user software license information. Responsive to receiving a request, the server locates license information for the user, and determines whether it needs to be updated. Updating a user's license information can mean terminating the license when unauthorized use is detected. For example, pass along piracy can be detected when the license verification information sent by the client indicates that the software has been copied to an unlicensed computer. However, updating a user's license can also involve other types of changes, for example extending, upgrading, restricting or downgraded the user's software license.

[0011] When the server determines that the user's license needs to be updated, the server stores the updated license and returns it to the client. The client receives the updated software license, and processes it as appropriate.

[0012] Because the client and the server continue to exchange software license information as the user operates the software program over time, the user's software license can be updated dynamically as desired. Updating the user's software license dynamically not only allows the termination or restriction of the license when fraud is detected, but also allows the modification of an active license, for example when the user purchases or declines various upgrades.

[0013] The features and advantages described in this summary and the following detailed description are not all-inclusive, and particularly, many additional features and advantages will be apparent to one of ordinary skill in the art in view of the drawings, specification, and claims hereof. Moreover, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter, resort to the claims being necessary to determine such inventive subject matter.

Brief Description of the Drawings

[0014] Figure 1 is a block diagram providing a high level overview of a system for dynamically managing software license information, according to some embodiments of the present invention.

[0015] Figure 2 is a flowchart, illustrating steps for a server to dynamically manage a user software license, according to one embodiment of the present invention.

[0016] Figures 3 and 4 are flowcharts, illustrating steps for a server to determine and process current software license information, according to some embodiments of the present invention.

[0017] Figure 5 is a flowchart illustrating steps for the server to modify the user software license information, according to one embodiment of the present invention.

[0018] Figure 6 is a flowchart illustrating server side steps for performing an embodiment of the present invention in which the server determines that the user is not licensed to run the software program.

[0019] Figure 7 is a block diagram illustrating one embodiment of the present invention in which license verification information comprises a hardware configuration identifier.

[0020] Figure 8 is a flowchart illustrating client side steps for performing an embodiment of the present invention in which the server determines that the user is not licensed to run the software program.

[0021] The figures depict embodiments of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention described herein.

Detailed Description

[0022] Figure 1 provides a high level overview of a system 100 for dynamically managing software license information 101 according to some embodiments of the present invention. A user 102 runs a client 103 software program on a computer

104. As the user 102 runs the software program 103, the user 102 attempts to access features 105 of the software program. Some features 105 are provided by a server 108, in which case the client 103 makes requests 107 to the server 108 to access the features 105. Other features are provided locally by the client 103 without the need to access the server 108. In some embodiments of the present invention, the client 103 is configured such that the user 102 attempting to access a client 103 provided feature 105 triggers a request 107 to the server 108, such that the server 108 can dynamically configure the user's software license information 101 as described herein. Which features 105 of the software program result in such request is a design choice, which can vary from embodiment to embodiment as desired.

[0023] Each such request 107 includes identification information 109 concerning the user 102, and license verification information 110 concerning the user 102. The format of the user identification information 109 can vary from embodiment to embodiment. In some embodiments, the user identification information 109 comprises a unique identifier provided by the software vendor with the user's copy of the software. Various formats of user identification information 109 will be readily apparent to one of ordinary skill in the relevant art. The format of the license verification information 110 can also vary from embodiment to embodiment. In one embodiment, license verification information 110 comprises a license file corresponding to the software program. In other embodiments, license verification information 110 can comprise sections of the license file, or other data that can be used to verify that the user is licensed to run the software program. Some examples are discussed in greater detail later in this specification.

[0024] As used herein, the term client 103 simply denotes those aspects of the software program associated with a user's computer 104, as well as underlying operating system and hardware support. As will be understood by those of skill in the art, a client 103 within the context of the present invention can comprise components of the software program, as well as components of the operating system of a user's computer 104 and hardware components of a user's computer 104.

[0025] As used herein, the term server 108 simply denotes those aspects of the software program associated with a remote computer 104, as well as underlying operating system and hardware support. As will be understood by those of skill in the art, a server 108 within the context of the present invention can comprise components of the software program, as well as components of the operating system of a remote computer 104 and hardware components of a remote computer 104.

[0026] The server 108 receives requests 107 from the client 103 to access features 105 of the software program. Although Figure 1 illustrates two requests 107 being transmitted by a client 103 to a server 108, the number two is of course only an example. One of ordinary skill in the relevant art will readily understand that over a period of time, the client 103 can make a variable number of requests 107 as desired.

[0027] Responsive to receiving a request 107, the server 108 retrieves stored software license information 101 concerning the user 102. The server 108 can utilize the user identification information 109 to locate stored software license information 101 concerning the user 102 in a manner that will be readily apparent to one of ordinary skill in the relevant art.

[0028] The server 108 determines current software license information 101 for the user 102. Determining current software license information 101 for the user 102 can comprise utilizing received license verification information 110 to determine whether the user 102 that transmitted the request 107 is licensed to run the software program, or to access the specific requested feature 105. Determining current license information can also comprise utilizing the license verification information 110 and/or the stored license information 101 to determine that the stored license information 101 is current, or that the stored license information 101 needs to be modified. Additionally, in some embodiments the request 107 can include verifiable directives from the client 103 to the server 108 to modify the user's software license information 101, such as a directive to upgrade the user's license accompanied by credit card information. Determining a user's current software license information 101 is discussed in greater detail later in this specification.

[0029] The server 108 returns the current software license information 101 for the user 102 to the client 103. Where the user 102 is authorized to access the requested feature 105 and the feature 105 is one provided by the server 108, the server 108 returns the feature 105 as well. Accessing and returning requested features 105 is discussed further below. The client's 103 processing of the received current software license information 101 is also discussed later in this specification.

[0030] Figure 2 illustrates steps for a server to dynamically manage a user software license, according to one embodiment of the present invention. A server 108 receives 201 a plurality of requests 107 from a client 103 to access at least one feature 105 of a software program. Each request 107 includes user identification information 109 and user license verification information 110.

[0031] Responsive to receiving a request 107, the server 108 retrieves 203 stored software license information 101 concerning the user 102. The server can utilize the received user identification information 109 to locate the software license information 101 corresponding to the user 102.

[0032] The server 108 proceeds to determine 205 current software license information 101 concerning the user 102. The server 108 then returns 207 the current software license information 101 concerning the user 102 to the client 103.

[0033] Figures 3 and 4 illustrate steps for the server 108 to determine 205 and process current software license information, according to some embodiments of the present invention. In the embodiment illustrated in Figure 3, the server determines 301 that the user 102 is not licensed to run the software program. The server 108 can make this determination in various ways. For example, in some embodiments the server 108 can examine the received license verification information 110, and determine that it does not match the stored license information 101 for the user 102. In other embodiments, the server 108 can determine from the received license verification information 110 and/or from the retrieved software license information 101 that the user's license to operate the software program has expired, or that the user 102 is trying to operate a pirated copy of the software program. Various alternatives will be apparent to those ordinary skill in the relevant art, some of which are discussed in greater detail later in this specification.

[0034] Turning to Figure 3, a server 108 receives 201 a request 107 from a client 103 to access a feature 105 of a software program, the request 107 including user identification information 109 and user license verification information 110. Responsive to receiving the request 107, the server 108 retrieves 203 stored software license

information 101 concerning the user 102. The server 108 proceeds to determine 301 that the user 102 is not licensed to run the software program. The server 108 then returns 303 the current software license information 101 to the client 103, the current software license information 101 indicating that the user 102 is not licensed to run the software program. The client's 103 processing of this current software license information 101 is discussed below.

[0035] The embodiment illustrated in Figure 4 is similar to the embodiment illustrated in Figure 3, except that in the embodiment illustrated in Figure 4 the server 108 determining 205 current software license information 101 concerning the user 102 comprises the server determining 301 that the user 102 *is* licensed to run the software program. A server 108 receives 201 a request 107 from a client 103 to access a feature 105 of a software program, the request 107 including user identification information 109 and user license verification information 110. Responsive to receiving the request 107, the server 108 retrieves 203 stored software license information 101 concerning the user 102. The server 108 proceeds to determine 401 that the user 102 is licensed to run the software program. The server 108 then returns 403 current software license information 101 to the client 103 indicating that the user 102 is licensed to run the software program. The client processes this current software license information 101 as discussed below.

[0036] Figure 5 illustrates steps for performing an embodiment of the invention in which the server 108 determining 205 current user software license information 101 comprises the server determining 501 that it is appropriate to modify the user software license information 101. As will be apparent to one of ordinary skill in the relevant art, there are a number of possible ways in which it can be appropriate to modify

a user's software license information 101. For example, if a user 102 is operating the software program for the first time, it could be desirable to activate the software license for the user 102. If a user's license has expired, or if piracy is detected, it could be desirable to deactivate the software license for the user 102. If the user 102 has purchased or otherwise arranged a license to use the software program for an extended period of time or on additional computers 104, extending the user software license could be in order. On the other hand, if the server detects that the user 102 arranged to license the software for less time or on fewer computers 104 than the current license information 101 indicates, it could be appropriate for the server 108 to restrict the user's license. Additionally, if the user has arranged for licensed access of more (or fewer) features of the software program, it could be appropriate for the server 108 to upgrade (or downgrade) the user's software license.

[0037] In the embodiment illustrated in Figure 5, as in the embodiments illustrated in Figures 2-4, a server 108 receives 201 a request 107 from a client 103 to access a feature 105 of a software program, the request 107 including user identification information 109 and user license verification information 110. Responsive to receiving the request 107, the server 108 retrieves 203 stored software license information 101 concerning the user 102.

[0038] In the embodiment illustrated in Figure 5, the server 108 proceeds to determine 501 that it is appropriate to modify the software license information 101 concerning the user 102. The server 108 can make such a determination in a number of ways. In some embodiments, the received license verification information 110, either on its own or when compared to the stored license information 101, indicates that the user

102 is not licensed to operate the software at all, or that the user's license should be restricted or downgraded. Comparing received license verification information 110 to the stored license information 101 is explained in greater detail below. In other embodiments, the server 108 receives a directive from the client 103 (either in the form of the license verification information 110 or another format) indicating that the user purchased or arranged for an expanded or upgraded license. Such directives must be verifiable by the server 108, for example by including valid credit card information for an upgrade purchase, or a unique identifier provided by the software vendor.

[0039] It is to be understood that the server 108 can also modify the user software license information 101 at the directive of the licensor, as opposed to the licensee or the licensee's client software program 103. In such instances, the licensor can input or transmit an appropriate directive to the server 108, which can then update the user software license information 101 without necessarily receiving a request 107 from the client 103 at all. In such instances, the server 103 proceeds to forward the updated software license information 101 to the client 103 with its response to the next client 103 request 107 (the server 108 does not update its stored license information 101 responsive to that client 103 request 107, in this case).

[0040] Responsive to the determination, the server 108 modifies 503 the software license information 101 concerning the user 102, and stores 505 the modified information 101. The server 108 also returns the modified software license information 101 concerning the user 102 to the client.

[0041] Figure 6 illustrates steps for performing an embodiment of the present invention in which determining that it is appropriate to modify software license

information 101 concerning the user 102 comprises determining that the user 102 is not licensed to run the software program. As in the embodiments illustrated in Figures 2-5, a server 108 receives 201 a request 107 from a client 103 to access a feature 105 of a software program, the request 107 including user identification information 109 and user license verification information 110. Responsive to receiving the request 107, the server 108 retrieves 203 stored software license information 101 concerning the user 102.

[0042] In the embodiment illustrated in Figure 6, the server 108 determines 601 that the user 102 is not licensed to run the software program. The server 108 can make this determination, for example, by examining the received license verification information 110 concerning the user 102, and determining the received license verification information 110 is non-authentic, or does not match the server's stored license information 101 concerning the user 102.

[0043] Responsive to the determination, the server deactivates 603 the user's software license, stores 605 the deactivated license information 101 concerning the user 102, and returns 607 the deactivated license information 101 to the client 103.

[0044] Returning to Figure 1, in some embodiments the server 108 determining current software license information 101 concerning the user 102 can comprise the server 108 determining that the user 102 is licensed to access a requested feature 105 of the software program. As illustrated in Figure 1, responsive to such a determination the server 108 provides the requested feature 105 of the software program to the client 103, where the feature 105 in question is one provided by the server 108.

[0045] The server 108 determining current software license information 101 concerning the user 102 can also comprise determining that the user 102 is not licensed to

access a requested feature 105 of the software program. When the server 108 so determines, the server 108 does not provide the requested feature 105 of the software program to the client 103, even where the feature 105 in question is one generally provided by the server 108.

[0046] Figure 7 illustrates an embodiment of the present invention in which the received license verification information 110 comprises a hardware configuration identifier 701 concerning the user's computer 104. Using techniques that will be readily apparent to one of ordinary skill in the art, the client 103 creates a hardware configuration identifier 701 that corresponds to the hardware configuration of the user's computer 104. The hardware configuration identifier 701 need not be a detailed listing of the complete hardware configuration of the user's computer 104, but contains information representing a sufficient number of hardware components to identify a computer 104 by its hardware configuration. One of ordinary skill in the relevant art will readily understand that various formats are possible for the hardware configuration identifier 701, for example a hash of selected components. The selection and number of components to include, as well as the format, are design choices.

[0047] In the embodiment illustrated in Figure 7, the client 103 includes a current hardware configuration identifier 701 with each request. Additionally, the stored license information 101 concerning the user 102 includes a hardware configuration identifier 701 concerning the user's computer 104. In this embodiment, the user's hardware configuration identifier 701 is stored by the server when the user's software license is initially activated. Subsequently, the stored hardware configuration identifier 701 can be updated, as explained below.

[0048] When the server 108 receives a request 107, the server 108 determines current software license information 101 concerning the user 102 by comparing the received hardware configuration identifier 701 to the stored hardware configuration identifier 701, to determine whether the received hardware configuration identifier 701 is acceptably similar to the stored hardware configuration identifier 701. Recall that when the user's software license was activated, the server 108 stored what was at that time a current identifier 701 of the hardware configuration of the user's computer 104. The received request 107 includes an identifier 701 of the hardware configuration of the user's computer 104 at the time the request 107 was transmitted. By comparing the stored hardware configuration identifier 701 to the received hardware configuration identifier 701, the server can determine whether or not the software program is being run on the licensed computer 104. If the received hardware configuration identifier 701 is acceptably similar to the stored hardware configuration identifier 701, the server determines that the request 107 was generated by the licensed computer 104. On the other hand, if the two identifiers 701 are not acceptably similar, the server determines that the request 107 was generated by a computer 104 other than the licensed computer 104. It is to be understood that the definition of "acceptably similar" is a design choice, which is a function of to what extent hardware modifications will be permitted without triggering a determination of a new computer 104. In one embodiment, the identifiers 701 must be identical to be considered acceptably similar, and in other embodiments varying amounts of difference are tolerated, as desired. For example, in some embodiments a specific number of hardware components must be identical for the two identifiers 701 to be considered acceptably similar.

[0049] In one embodiment illustrated by Figure 7, when the server 108 determines that the received hardware configuration identifier 701 is acceptably similar but not identical to the stored hardware configuration identifier 709 then the server 108 returns its stored hardware configuration identifier 701 to the client as part of the license information 101. Note that the hardware configuration identifier 701 returned by the server may be different from the current client hardware configuration identifier 701.

[0050] When the received hardware configuration identifier 701 is not acceptably similar to the stored hardware configuration identifier 701, the server determines that the user 102 is not licensed to run the software program. Because the request 107 came from an unlicensed computer 104, the server determines the user operating the software program must be unlicensed. The server 108 updates the stored software license information 101 concerning the user 102 to indicate that the user is not licensed to run the software program. The server 108 also returns software license information 101 indicating that the user is not licensed to run the software program to the client 103.

[0051] Returning our attention to Figure 1, this specification will now explain client 108 side processing according to some embodiments of the present invention. In some embodiments, a client 108 sends a plurality of requests 107 to a server 108 to access features 105 of a software program, each request 107 including identification information 109 concerning the user 102, and license verification information 110 concerning the user 102. In response to a request 107, the client receives current software license information 101 concerning the user 102 from the server 108. If the server 108 determines that the user 102 is licensed to access the requested feature

105, the client receives the requested feature 105 as well, where the feature 105 in question is one provided by the server 108. The client 103 proceeds to store the received current software license information 101 concerning the user 102. Thus, each time the client 103 attempts to access a feature of the software program, the client 103 receives current software license information 101 concerning the user 102 from the server 108.

[0052] Recall that the server 108 can modify the license information 101 concerning the user 102. When this occurs, the current software license information 101 returned by the server to the client 103 comprises modified license information 101. The modified license information 101 can comprise any of the modifications discussed above, e.g., an activated user software license, a deactivated user software license, an extended user software license, a restricted user software license, an upgraded user software license or a downgraded user software license.

[0053] Returning to Figure 7, note that the client 103 can store the current software license information 101, including hardware configuration information 701. In some embodiments, the stored current software license information 101 can also include expiration dates for some or all features. In some embodiments, the stored software license information 101 is utilized to allow the client 103 to run without communicating with the server 108 every time it needs a feature. In these embodiments, the client 103 checks the stored software license information 101 to confirm that the user 102 is authorized to run the feature, for example by ensuring that the local license information contains an indication that the feature is enabled, not expired, and/or that the hardware configuration identifier 701 stored on the client is sufficiently similar to the hardware configuration identifier generated for that client 103, as described above. In some

embodiments, the ability of a client 103 to run without communicating with the server 108 can be restricted as desired, e.g., by limiting the number of days or the number of times that the software is allowed to run on local license information, without communicating with the server 108.

[0054] Figure 8 illustrates client 103 side steps for performing an embodiment of the present invention in which the user 102 is not licensed to run the software program. As explained above, sometimes the server 108 will determine that the user 102 is not licensed to run the software program. When this occurs, the current software license information 101 concerning the user 102 received 801 by the client 103 indicates that the user 102 is not licensed to run the software program. In response, in some embodiments the client 103 terminates 803 the software program, such that the user 102 cannot run the software program. In other embodiments, the client 103 responds in other ways as desired. For example, the client 103 can display a warning to the user 102 indicating that the user is not licensed to run the software program. The client 103 can provide the user 102 with an opportunity to purchase a license, for example by entering credit card information. The client 103 can allow a certain number of unlicensed grace uses, or allow unlicensed use for a specific period of time. As will be apparent to one of ordinary skill in the relevant art, various possible client 103 responses to detected unlicensed use of the software program are possible, all of which are within the scope of the present invention.

[0055] Returning to Figure 7, as explained above, in some embodiments the current software license information 101 that the server 108 returns to the client 103 includes a hardware configuration identifier 701 of a computer 104 associated with the

user 102. In such embodiments, the client 103 uses techniques that will be readily apparent to one of ordinary skill in the art to determine the current hardware configuration of the user's computer 104, and to create a corresponding current hardware configuration identifier 701. The client 103 then compares the received hardware configuration identifier 701 to the current hardware configuration identifier 701.

[0056] Responsive to the received hardware configuration identifier 701 being acceptably similar to the current hardware configuration identifier 701, the client 103 allows the user 102 to run the software program. On the other hand, if the received hardware configuration identifier 701 is not acceptably similar to the current hardware configuration identifier 701, the client 103 terminates the software program, such that the user 102 cannot run the software program. Alternatively, the client can respond in other ways as desired. For example, the client 103 can display a warning to the user 102 indicating that the user is not licensed to run the software program, provide the user 102 with an opportunity to purchase a license, allow a certain number of unlicensed grace uses, and/or allow unlicensed use for a specific period of time.

[0057] As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Likewise, the particular naming and division of the modules, features, attributes, methodologies, clients, servers and other aspects are not mandatory or significant, and the mechanisms that implement the invention or its features may have different names, divisions and/or formats. Furthermore, as will be apparent to one of ordinary skill in the relevant art, the modules, features, attributes, methodologies, clients, servers and other aspects of the invention can be implemented as software, hardware,

firmware or any combination of the three. Of course, wherever a component of the present invention is implemented as software, the component can be implemented as a standalone program, as part of a larger program, as a plurality of separate programs, as a statically or dynamically linked library, as a kernel loadable module, as a device driver, and/or in every and any other way known now or in the future to those of skill in the art of computer programming. The clients and servers discussed herein can each be implemented on a single computing device or on multiple computing devices as desired. Wherever functionality is described as being performed by a client or by a server, it is to be understood that the functionality can be provided by one or more components, and that these components can have other names as desired. Additionally, the present invention is in no way limited to implementation in any specific programming language, or for any specific operating system or environment. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.